title: Sending Bundles

import Tabs from '@theme/Tabs' import TabItem from '@theme/TabItem' import SendBundleRpc from '@site/docs/specs/mev-share/_mev_sendBundle.mdx' import BuilderInheritance from '@site/docs/specs/mev-share/blurbs/_builderInheritance.mdx'

While the new bundle spec may look complex, a bundle can be sent with default parameters using a client library, which ends up looking quite similar to sending bundles on mev-boost.

Sending a Bundle

:::info Placeholders

Variables in ALL_CAPS are placeholders. To use this code, replace them with your own data.

:::

```typescript
import MevShareClient, { BundleParams } from '@flashbots/mev-share-client'

const mevShareClient = MevShareClient.useEthereumMainnet(authSigner)

// ...

const bundle = [ {hash: PENDING_TX_HASH}, {tx: await wallet.signTransaction(BACKRUN_TX), canRevert: false}, ]

const params: BundleParams = { inclusion: { block: targetBlock, // target several blocks with maxBlock maxBlock: targetBlock + 3, }, body: bundle, }

const bundleResult = await mevShareClient.sendBundle(params)
```

The first transaction in the bundle contains the hash of a pending transaction, which we presumably got from listening to the event stream. The second transaction is one we sign ourself, which backruns the first transaction, presumably for a profit.

:::info Bundle endpoint

MEV-Share bundles are sent to the same RPC endpoint as MEV-Boost bundles: https://relay.flashbots.net

:::

See mev-share-client-ts for a full working example.

The params not included in the previous example can be used to configure more advanced conditions for your bundle's execution and the MEV refunds you earn.

Share bundle data

To earn more MEV refunds and improve your bundle's chances of inclusion, you can specify data about your bundle's transactions that you wish to share with other searchers. The data that is shared is specified via the privacy parameter.

:::caution bundles with signed txs only

Note that bundles with transactions including the hash parameter are considered "unmatched", and will throw an error.

:::

```typescript
const params: BundleParams = { inclusion: { block: 17539448, maxBlock: 17539450 }, body: [ {tx: "0x02...", canRevert: false}, {tx: "0x02...", canRevert: false}, ], privacy: { hints: { calldata: true, logs: true, txHash: true, }, }, }
```

```bash
curl -X POST -H "Content-Type: application/json" -H "X-Flashbots-Signature: $FB_SIG" -d '{ inclusion: { block: "0x10ba178", maxBlock: "0x10ba17a", }, body: [ {tx: "0x02...", canRevert: false}, {tx: "0x02...", canRevert: false}, ], privacy: { hints: ["calldata", "logs", "tx_hash"], }, }'
```

Even sharing only the tx_hash hint can be enough to earn some extra MEV. The only requirement to earn it is that your bundle is included in another searcher's bundle.

This strategy is particularly relevant to searchers who operate on public mempool transactions, and/or employ strategies that leave leftover MEV on the table (for example, if your strategy changes prices enough for a backrun arbitrage to be profitable).

Nest bundles (be a MEV-Share Node)

MEV-Share Nodes nest bundles to build composite bundles that are more profitable.

```typescript
const params: BundleParams = {
  inclusion: {
    block: TARGET_BLOCK,
  },
  body: {
    {hash: PENDING_TX_HASH},
    {bundle: {
      inclusion: {...},
      body: {...},
      privacy: {...},
      validity: {...},
    }}
  },
}
```

Builder Inheritance

Simulating Bundles

Bundles matched by the MEV-Share Node can be simulated to check MEV profits. To simulate a bundle, we use the [mev_simBundle](#) endpoint.

:::caution only matched bundles can be simulated

Bundle simulations can only be executed on matched bundles, which contain only signed transactions, or nested bundles which also contain only signed transactions. Bundles with transactions including the `hash` parameter are considered "unmatched", and will throw an error.

:::

```typescript
const params: BundleParams = {
  inclusion: { block: TARGET_BLOCK },
  body: [
    {bundle: {
      version: "beta-1",
      inclusion: {
        block: TARGET_BLOCK,
        maxBlock: TARGET_BLOCK + 3,
      },
      body: [{tx: SIGNED_TX_1}],
      privacy: { hints: {calldata: true}, },
      validity: { refundConfig: [{address: REFUND_ADDRESS, percent: 100}] }
    }},
    {tx: SIGNED_TX_2},
  ],
  validity: { refund: [{bodyIdx: 0, percent: 90}] }
}

const simResult = await mevShareClient.simulateBundle(params)
```